Name: **Student id:** **Sect#:** **#:**

QUESTION #	1	2	3	4	5	TOTAL
MAX POINTS	12	12	15	15	11	65
POINTS EARNED						

****

UNIVERSITY OF BAHRAIN **COLLEGE OF INFORMATION TECHNOLOGY**
DEPARTMENT OF COMPUTER SCIENCE

ITCS 242: ASSEMBLY LANGUAGE PROGRAMMING

1st SEMESTER 2015/2016 **DATE: NOV 11, 2015** **FIRST TEST**

****

QUESTION ONE: *Show your work!* *{12 points}*

1) Using 16 bits to represent signed values, the smallest decimal value is -2^{15} and the largest decimal value is $+2^{15} - 1$

2) Perform the following operation using 16's complement: 4F2A – 7C9E.

4F2A – 7C9E = 4F2A + (–7C9E)

= 4F2A + 8362 = d28C

3) The 8-bit value 11100010 represents unsigned decimal value **226** and signed decimal value **-30**.

4) Using 8 bits to represent numbers, convert the value $(+99)_{10}$ into binary **0110 0011** and the value $(-99)_{10}$ **9D** into hexadecimal.

5) Perform the following operation: *3A8H + 2764Q*= (**9 9 C**) H.

```
0011 1010 1000
0101 1111 0100
1001 1001 1100
  9    9    C
```

6) Using 8 bits to store numbers, show how the operation $(29)_{10} - (77)_{10}$ is performed by the computer.

```
+29 = 0001 1101          0001 1101
+77 = 0100 1101    →   + 1011 0011
-77 = 1011 0011          1101 0000
```

QUESTION TWO: ***Fill in blanks*** *{12 points}*

1) A logical address consists of two values: **segment** and **offset** values.
2) The two kinds of labels used in Assembly programs are: **data labels** and **code labels**.
3) Name two main components in the CPU: **registers** and **arithmetic/logical unit**.
4) In the instruction: `MOV beta, BX;` the type of the destination operand is **direct.** The type of the source operand in the instruction `SUB ECX, [EBX];` is **indirect**.
5) In real-address mode, the maximum main memory size is **1 MB**. In protected mode, the maximum main memory size is **4 GB**.
6) The address of the next instruction is stored in **program counter register**, and the fetched instruction is stored in **instruction register**.
7) CPU registers are accessed by **names** and main memory locations are accessed by **addresses**.
8) The addresses used by the hardware are called **absolute**. The memory table used to store the starting addresses of the active segments is called **segment descriptor table**.
9) The symbolic addresses are translated into logical address by **compilers/Assemblers**. The logical addresses are translated into absolute addresses by a **loader**.
10) The flag that indicates whether the instruction result is negative or not is **sign.** The flag that indicates whether the instruction result contains odd or even number of ones is **Parity**.

11) If a computer has 16 GB of main memory, the minimum number of address lines is **34**.

12) If the physical address is 60000H and the offset value is 2F90H, then the segment value will be: **5D07**H

QUESTION THREE: *Write a complete assembly program that:* *{15 points}*

- defines an array TAR consisting of 24 elements of signed words.
- fills the array TAR with values by randomly generating 48 values in the range ***(-780 ... +250)***
- displays in HEX all elements of array TAR as double words separated by a space.
- displays in signed decimal all elements of array TAR separated by tabs.
- stores -6 in the low-order byte and +6 in the high-order byte of each element of array TAR.

```
            INCLUDE  Irvine32.inc
            .DATA
      TAR   SWORD    24 dup(?)

            .CODE
      MAIN  PROC
            CALL     RANDOMIZE
; Generating random numbers and storing them in array TAR
            MOV      ECX, LENGTHOF TAR
            MOV      ESI, 0                ; INDEX
      L0:   MOV      EAX, 1031
            CALL     RANDOMRANGE
            SUB      EAX, 780
            MOV      TAR[2*ESI], ax
            INC      ESI
            LOOP     L0
            CALL     CRLF
; Display elements of array TAR as dwords (HEX) separated by space
            LEA      ESI, TAR
            MOV      EBX, TYPE TAR * 2
            MOV      ECX, LENGTHOF TAR / 2
            CALL     DUMPMEM
            CALL     CRLF
; Display all elements of TAR in signed decimal separated by TABs
            MOV      ECX, LENGTHOF TAR
            MOV      ESI, 0
      L2:   MOVSX    EAX, TAR[2*ESI]
            CALL     WRITEINT
            MOV      AL, 9             ; ASCII OF TAB IS 9
            CALL     WRITECHAR
            INC      ESI
            LOOP     L2
; stores -6 in the low-order byte and +6 in the high-order byte
            MOV      ESI, 0
            MOV      ECX, LENGTHOF TAR
      L9:   MOV      TAR[2*ESI], 06FAH
            INC      ESI
            LOOP     L9

            EXIT
      MAIN  ENDP
            END      MAIN
```

QUESTION FOUR: {15 points}

(a) Given the following data definitions, write the needed instructions to prompt the user to enter from the keyboard up to 120 characters and store them in an array named **urarr**. Display the entered characters at the beginning of a new line in reverse order.

```
msg    byte     "Enter up to 120 characters please: ",0
urarr  byte     121 dup (?), 0

       lea      edx, msg
       call     writestring

       mov      ecx, 120
       lea      edx, urarr
       call     readstring

       call     crlf
       mov      ecx, eax
       dec      ecx
L8:    mov      aL, urarr[ecx]
       call     writechar
       loop     L8
```

(b) Given: `ARR SWORD 3A78H,87CFH,...;` Write the needed instructions to swap the two bytes in each word of array `ARR`.

```
       mov      ecx, lengthof arr
       mov      esi, 0

L9:    mov      ah, byte ptr arr[2*esi]
       xchg     ah, byte ptr arr[2*esi+1]
       mov      byte ptr arr[2*esi], ah
       inc      esi
       loop     L9
```

(c) Write the needed instructions to display as dwords the contents of `ARR` in binary one value per line.

```
       mov      ecx, lengthof arr/2
       mov      ebx, 0
L6:    call     crlf
       mov      eax, dword ptr arr[4*ebx]
       call     writebin
       inc      ebx
       loop     L6
```

QUESTION FIVE: [11 pts]

Carefully study the following Assembly code, and then answer the questions in parts: 1 and 2

```
UT    DWORD     725A9033H, 5C6F3A49H, 69CB3A2CH, 248F7C39H
T1    SBYTE     9AH, 22H, 9CH, 0C4H, "ABFC6789", 5EH, 3BH, 80H, 1FH
T2    WORD      6F7FH, 6ACAH, 81CFH, 69CFH, 3459H, ?, ?, 7CH
rat   _____________________________________

      MOV       AX, WORD PTR UT+6
      MOV       BX, WORD PTR T2-2
      MOV       CL, SIZEOF T2
      MOV       CH, TYPE UT*4
      MOV       DX, WORD PTR T1
      MOVSX     DI, T1[3]
```

Part#1: Answer each of the following 3 questions as required:

1) The instruction that makes EBX point to the second element in UT is ***LEA EBX, UT[4]***
2) The instruction that swaps the last word of T2 with CX is ***XCHG CX, T2[sizeof T2-2]***
3) The instruction that stores in EAX the number of words in UT is ***MOV EAX, SIZEOF UT/2***.
4) The statement that defines a constant rat equals to the number of bytes defined in all UT, T1, and T2 directives is ***rat equ $-UT.***

Part#2: Execute the above instructions and answer each of the following 4 questions

5) The register DI will contain:
   a) 009CH b) 00C4H **c) FFC4H** d) FF9CH e) None

6) The register BX will contain:
   a) 803BH b) 801FH c) 3B80H **d) 1F80H** e) None

7) The register CX will contain:
   a) 1010H b) 1616H c) 1610H d) 1016H e) None

8) The register AX will contain:
   a) 3A49H **b) 5C6FH** c) 6F5CH d) 493AH e) None

Part#3: Answer each of the following questions as required:

9) The statement that produces syntax error during assembly process is:
   a) INC AX b) MOVZX EBX,CL c) ADD AX,BX **d) MOVSX BX,AX** e) None

10) The statement that produces syntax error during assembly process is:
   a) MOV AX,[EBX] b) MOVZX EBX, CL **c) MOV [EBX],[EAX]** d) None

11) The statement that produces syntax error during assembly process is:
   a) MOV AX,[EBX] b) XCHG BX, CX **c) INC [EAX]** d) MOV BX,[EAX] e) None